１．sample#1.c を解析し、ASCII コード(0x00ー0x7f)の各範囲(Scope)を判断するプログラムを作成せよ。
A.解析

#define FALSE 0
#define TRUE !FALSE
マクロ定義。
記号定数 FALSE に0を入れる。また記号定数 TRUE に0以外を入れる。

int value, c, count;
変数設定をしている。
value:使っていない。
c:
count:ループ回数を数えて、記憶させておく変数。
char line [128]:文字コードを格納する所（line という入れ物）を128個作成している
count = 0:count 変数に0を代入する。

while
  count++;
  if(count > 5) break;
変数countを使ってループ回数を数えさせている。
下の処理をするたびにcountに１足されていく。countの初期値が上で０と設定されているおり、５回ループした後、つまり６回目にbreakの役割によりループから抜けるようになっている。

printf("Enter a HexValue ===> ");
printfにより（）内の文字を表示させる。

fgets("line, sizeof(line),stdin");
fgetsにより変数lineに入力された値を保存させる。
sizeofは、（）内この場合は変数lineのデーター型のサイズをバイト数で返す演算子。
stdinは、標準入力の意味。

sscanf(line, "%x", &c);
sscanfは入力を意味している。ここではlineの中身を16進数の数値として、
変数cの中に代入させる。
（&は、値をポインタ変数に代入するために使用している）
“%x” により16進数の数値に変換されている。%dなら１０進数の数値に変換される。

 printf("Colum=%02d:%%d(%3d)-%%x(%2x)",count,c,c);
ループ回数を2桁の１０進数により表示、また変数cの値を3桁の10進数により表示させ
変数cの値を２桁の16進数で表示させている。
％０２dと指定すると数字２桁で出力し、空の部分は０を入れて表示させる。
例　　１→０１
％２dと指定すると数字２桁で出力し、空の部分はスペースを入れて表示させる。
例　　２→＿２（「＿」はスペースを表している）

if(0x20 <= c && c <= 0x7e)
     printf("-%%c(%c)\n",c);
   else
     printf("-Not Printable character\n");
if文を使って条件（）の中身の真偽に基づきプログラムを選択的に実行するための文
変数cの値が０x２０ー０x７eの間ならASCIIコードに対応した文字（表示可能な文字）を表示。
それ以外（表示できない文字）なら-Not Printable characterと表示させる。
＆＆は「０x２０≦c　∩（かつ）　c≦0x７e」が真である場合に真を返す演算子。
％cは変数を文字として表示させる。

```
   if     (0x20 <= c && c <= 0x2f){
     puts("====> Scope_A\n");
   }else if(0x30 <= c && c <= 0x39){
     puts("====> Scope_B\n");
   }else if(0x3a <= c && c <= 0x40){
     puts("====> Scope_C\n");
   }else if(0x41 <= c && c <= 0x5a){
     puts("====> Scope_D\n");
   }else{
     puts("====> Scope_E\n");
   }
 }
```

変数の中身にあわせて表示するものを変更させている。
20ー２fの間ならA、30ー39の間ならB、3aー４０の間ならC、４１ー５aの間ならD
それ以外だとEが表示される。

解析終了

１-２　　ASCIIコード(0x00ー0x7f)の各範囲(Scope)を判断するプログラムを作成せよ。
（範囲例）数字：figures、英大文字：capital letter、英小文字：small letter、
非表示文字：Not Printable character、その他：misc.等

【ソース】

```
０１        include <stdio.h>

０２        #define FALSE 0
０３        #define TRUE  !FALSE

０４        int main(){
０５     int c, count;
０６       char line[128];

０７                count = 2009;

０８                while(TRUE){
０９                count--;
10                 if(count < 1994) break;

１１              printf("16進数で入力 → ");
１２              fgets(line, sizeof(line), stdin);
１３              sscanf(line, "%x", &c);

１４              printf("Colum=%02d:%%d(%3d)-%%x(%2x)",count,c,c);
１５              if(0x21 <= c && c <= 0x7e)
１６           printf("-%%c(%c)\n",c);
１７              else
１８           printf("-Not Printable character\n");
１９              if     ( 0x00 <= c && c <= 0x20){
２０        puts("====> 非表示\n");
２１          }else if(c ==  0x21){
２２        puts("====> びっくり！\n");
２３          }else if((c == 0x24) || (c == 0x5c)){
２４        puts("====> お金の記号\n");
```

```
２５          }else if((c == 0x22) || (c == 0x23) || (c == 0x2c) || (c == 0x27) || (c == 0x3a) || (c == 0x3b) || (c == 0x3b) || (c == 0x5e) || (c == 0x5f) || (c == 0x60) || (c == 0x7c) || (c == 0x7e)){
２６          puts("====> その他の記号 \n");
２７          }else if((c == 0x25) || (c == 0x2a) || (c == 0x2b) || (c == 0x2d) || (c == 0x2e) || (c == 0x2f) || (c == 0x3d) || ( 0x30 <= c && c <= 0x39  )){
２８          puts("====> 計算機についてる物\n");
２９          }else if(c == 0x26){
３０          puts("====> AND\n");
３１          }else if((c == 0x3c) || (c == 0x3e)){
３２          puts("====> 不等号\n");
３３          }else if(c == 0x3f){
３４          puts("====> はてなマーク\n");
３５          }else if(c == 0x40){
３６          puts("====> アットマーク\n");
３７          }else if((c == 0x28) || (c == 0x29)){
３８          puts("====> 括弧\n");
３９          }else if((c == 0x7b) || (c == 0x7d )){
４０          puts("====> 中括弧\n");
４１          }else if((c == 0x5b) || (c == 0x5d )){
４２          puts("====> 大括弧\n");
４３          }else if(0x41 <= c && c <= 0x5a){
４４          puts("====> アルファベット大文字\n");
４５          }else if(0x61 <= c && c <= 0x7a){
４６          puts("====> アルファベット小文字\n");
４７          }else if(c == 0x7f ){
４８          puts("====> DEL\n");
４９          }else{
50          puts("====> 不明\n");
５１          }
５２  }
５３  return(0);
５４}
```

【結果】

```
Colum=2008:%d( 16)-%x(10)-Not Printable character
====> 非表示

16進数で入力 → 21
Colum=2007:%d( 33)-%x(21)-%c(!)
====> びっくり！

16進数で入力 → 24
Colum=2006:%d( 36)-%x(24)-%c($)
====> お金の記号

16進数で入力 → 2c
Colum=2005:%d( 44)-%x(2c)-%c(,)
====> その他の記号

16進数で入力 → 2b
Colum=2004:%d( 43)-%x(2b)-%c(+)
====> 計算機についてる物

16進数で入力 → 26
```

Colum=2003:%d(38)-%x(26)-%c(&)
====> AND

16進数で入力 → 3c
Colum=2002:%d(60)-%x(3c)-%c(<)
====> 不等号

16進数で入力 → 3f
Colum=2001:%d(63)-%x(3f)-%c(?)
====> はてなマーク

16進数で入力 → 40
Colum=2000:%d(64)-%x(40)-%c(@)
====> アットマーク

16進数で入力 → 28
Colum=1999:%d(40)-%x(28)-%c(()
====> 括弧

16進数で入力 → 7d
Colum=1998:%d(125)-%x(7d)-%c(})
====> 中括弧

16進数で入力 → 5b
Colum=1997:%d(91)-%x(5b)-%c([)
====> 大括弧

16進数で入力 → 67
Colum=1996:%d(103)-%x(67)-%c(g)
====> アルファベット小文字

16進数で入力 → 4e
Colum=1995:%d(78)-%x(4e)-%c(N)
====> アルファベット大文字

16進数で入力 → 7f
Colum=1994:%d(127)-%x(7f)-Not Printable character
====> DEL

16進数で入力 → ui
Colum=2008:%d(-1881139893)-%x(8fe0154b)-Not Printable character
====> 不明

解説
else if文を何個か追加して、いろいろ分類してみた。
「または」を意味する演算子「||」を使いソースを組んでみた。
それぞれのASCIIコードに対応する範囲は以下の通りである。
００ー２０　非表示
２１（！）　びっくり！
２２、２３、２７、2C、３A、３B、５E、５F、６０、７C、７E その他の記号
２４、５C（$）、(¥) お金記号
２５、２A、２B、２D、2E、２F、３０ー３９、３D　計算機についているもの
２６　（&）　アンド
２８、２９　括弧「()」
3C、3E 不等号
３F　（?）　はてなマーク

４０（@）アットマーク
４１ー５Ａアルファベット大文字
６１ー７Ａアルファベット小文字
７F DEL

考察
(c == 0x22) || (c == 0x23)とするところを(c == 0x22) | (c == 0x23)としたり、また(c == 0x22 | 0x23)、
(c == 0x22 || 0x23)としても同様な結果を導くことができる。

2. sample#2.c のプログラムの動作を考察せよ。

```
#define FALSE 0
#define TRUE !FALSE
```

マクロ定義。
FALSEに０を入れる。TRUEに０以外を入れる

```
int  count;
count = 0
```

変数countの作成
countの初期値を０とする。

```
while(TRUE){
  count++;
```

条件が偽（０）になるまで、while文の中身が実行される。
countの値を１増やす。

```
if(count > 5) break;
printf("While-Count=%2d\n",count);
```

whileを使って、1ー5までの数字を表示する
countの値が５より大きくなったときにbreakつまりループから脱出する。
countの値を2桁の10進数で表示させる。

```
 for(count=1; count<=5; count++){
 printf("for  -Count=%2d\n",count);
```

forを使って、1ー5の数字を表示する
変数countの初期値を1に設定。for内の処理を実行した後でcountに1を足す。
countが６以上になったらループを抜ける（break）。
上のwhileと処理は同じだが、whileだとprintfの前、forだとprintfの後に
countに1を足しているという違いがある。
whileでは初期値を0、forでは初期値を1としている。
よって内容はwhile文を使ったループで1~5までの数字を表示し、
次にfor文を用いたループで1~5までの数字を表示させるものである。

おまけ
Fibonacci数列をwhile文とfor文を使って作った
・while文

```
#include <stdio.h>
  int old_number;
  int current_number;
  int next_number;
  int n;
int main()
{
  old_number = 1;
  current_number = 1;
```

```
  n = 1;
  printf("F(0) = 0\n");
  printf("F(1) = 1\n");
  while (current_number < 1000){
    n++;
    printf("F(%d) = %d\n",n,current_number);
    next_number = current_number + old_number;
    old_number = current_number;
    current_number = next_number;
  }
  return(0);
}
```

・for文

```
#include <stdio.h>
  int old_number;
  int current_number;
  int next_number;
  int n;
int main()
{
  current_number = 1;
  n = 2;
  printf("F(0) = 0\n");
  printf("F(1) = 1\n");
  for(old_number = 1; current_number < 1000; n++)
  {
            printf("F(%d) = %d\n",n,current_number);
            next_number = current_number + old_number;
            old_number = current_number;
            current_number = next_number;
  }
  return(0);
```

結果

```
F(0) = 0
F(1) = 1
F(2) = 1
F(3) = 2
F(4) = 3
F(5) = 5
F(6) = 8
F(7) = 13
F(8) = 21
F(9) = 34
F(10) = 55
F(11) = 89
F(12) = 144
F(13) = 233
F(14) = 377
F(15) = 610
F(16) = 987
```

考察
for文を使う場合だとｎに＋1されるタイミングがwhile文のときよりも遅いので、
for文の場合はｎの値を2にしないと同じような結果にはならない。
while文の方が扱いが楽に感じた。

define(マクロ定義)とint(変数)
マクロと変数の違いを考察。
変数と同じように演算ができるかどうか試してみた。

変数でのソース

```
#include <stdio.h>
int FALSE;
int main(){
	FALSE = 0;
	while(FALSE < 5){
		FALSE++;
		printf("FALSE = %d\n",FALSE);
	}
	return(0);
}
```

結果

```
FALSE = 1
FALSE = 2
FALSE = 3
FALSE = 4
FALSE = 5
```

マクロ定義でのソース

```
#include<stdio.h>
#define FALSE 0
int main(){
	while(FALSE < 5){
		FALSE++;
		printf("FALSE = %d\n",FALSE);
	}
	return(0);
}
```

結果

```
sample5.c(5) : error C2105: '++' には左辺値が必要です。
```

考察
最初にFALSEが0であると定義したはずなのに左辺値が必要といわれることから
defineを変数として利用することは無理なようである。

3. sample#3.c を解析し、表示可能な文字によるASCIIコード表を作成せよ。

1-解析

```
int main () {
int c;
```

変数cの設定

```
for(c = 0x20 ; c < =0x40;  c++) {
if ( (c % 4)  == 0) printf(“\n”) ;
```

cの初期値は16進数における20。Cは40になるまでループし続ける。
41になるとループから脱出する。
cの値が4で割ったときにあまりが0の場合、改行される。

```
printf(“%%x(%x)-%%c(%c) ｜”,c,c);
```

printf("¥n");
%x(変数cの値)-%c(対応する文字)¦　の書式で表を作成。
プロンプト改行

以上より0x2Õ～0x40までのASCIIコードの文字を4列の表にするプログラムであることがわかる。

2-表示可能な文字によるASCIIコード表を作成。

```
#include <stdio.h>
int main(){
 int c;
    for(c = 0x20; c<=0x7e; c++){
    if((c % 4) == 0) printf("¥n");
    printf("%%x(%x)-%%c(%c) | ",c,c);
    }
    printf("¥n");
    return(0);
}
```

結果

%x(20)-%c() | %x(21)-%c(!) | %x(22)-%c(") | %x(23)-%c(#) |
%x(24)-%c($) | %x(25)-%c(%) | %x(26)-%c(&) | %x(27)-%c(') |
%x(28)-%c(() | %x(29)-%c()) | %x(2a)-%c(*) | %x(2b)-%c(+) |
%x(2c)-%c(,) | %x(2d)-%c(-) | %x(2e)-%c(.) | %x(2f)-%c(/) |
%x(30)-%c(0) | %x(31)-%c(1) |%x(32)-%c(2) | %x(33)-%c(3) |
%x(34)-%c(4) | %x(35)-%c(5) | %x(36)-%c(6) | %x(37)-%c(7) |
%x(38)-%c(8) | %x(39)-%c(9) | %x(3a)-%c(:) | %x(3b)-%c(;) |
%x(3c)-%c(<) | %x(3d)-%c(=) | %x(3e)-%c(>) | %x(3f)-%c(?) |
%x(40)-%c(@) | %x(41)-%c(A) | %x(42)-%c(B) | %x(43)-%c(C) |
%x(44)-%c(D) | %x(45)-%c(E) | %x(46)-%c(F) | %x(47)-%c(G) |
%x(48)-%c(H) | %x(49)-%c(I) | %x(4a)-%c(J) | %x(4b)-%c(K) |
%x(4c)-%c(L) | %x(4d)-%c(M) | %x(4e)-%c(N) | %x(4f)-%c(O) |
%x(50)-%c(P) | %x(51)-%c(Q) | %x(52)-%c(R) | %x(53)-%c(S) |
%x(54)-%c(T) | %x(55)-%c(U) | %x(56)-%c(V) | %x(57)-%c(W) |
%x(58)-%c(X) | %x(59)-%c(Y) | %x(5a)-%c(Z) | %x(5b)-%c([) |
x%x(5c)-%c(¥) | %x(5d)-%c(]) | %x(5e)-%c(^) | %x(5f)-%c(_) |
%x(60)-%c(`) | %x(61)-%c(a) | %x(62)-%c(b) | %x(63)-%c(c) |
%x(64)-%c(d) | %x(65)-%c(e) | %x(66)-%c (f) | %x(67)-%c(g) |
%x(68)-%c(h) | %x(69)-%c(i) | %x(6a)-%c(j) | %x(6b)-%c(k) |
%x(6c)-%c(l) | %x(6d)-%c(m) | %x(6e)-%c(n) | %x(6f)-%c(o) |
%x(70)-%c(p) | %x(71)-%c(q) | %x(72)-%c(r) | %x(73)-%c(s) |
%x(74)-%c(t) | %x(75)-%c(u) | %x(76)-%c(v) | %x(77)-%c(w) |
%x(78)-%c(x) | %x(79)-%c(y) | %x(7a)-%c(z) | %x(7b)-%c({) |
%x(7c)-%c(¦) | %x(7d)-%c(}) | %x(7e)-%c(~) |

考察
「cが4の倍数なら、cに対応する文字を表示する前に改行する」ので、最初の行が1行空いている。

4.文字（文字列では無い）の演算について考察せよ。例）('a'-'A')?、('f'-'a')?

```
#include<stdio.h>
int main(){
int answer;
answer ='a' - 'A';
printf("a(0x%x)-A(0x%x)=",'a','A');
printf("%c(0x%x)\n",answer,answer);
```

```
answer = 'f' - 'a';
printf("f(0x%x)-a(0x%x)=",'f','a');
printf("%c(0x%x)\n",answer,answer);

answer = 'z' - 0x20;
printf("z(0x%x)-20(0x20) = ",'z');
printf("%c(0x%x)\n",answer,answer);

return(0);
}
```

結果

```
a(0x61)-A(0x41)= (0x20)
f(0x66)-a(0x61)=#(0x5)
z(0x7a)-20(0x20) = Z(0x5a)
```

考察

とりあえず例題の'a'-'A','f'-'a'をやってみた。
同じアルファベットの大文字から小文字を引く0x20となることから
大文字を小文字に変換するプログラムを組めそうである。

5. エラー報告

error1

ASCIIコード(0x00ー0x7f)の各範囲(Scope)を判断するプログラムのところで
else if(c == 0x21)としなければ、ならないところをelse if(c = 0x21)とすると、
下記のようにすべてびっくり！の範囲になってしまう。

```
16進数で入力 → 5f
Colum=2007:%d( 95)-%x(5f)-%c(_)
====> びっくり！

16進数で入力 → 67
Colum=2006:%d(103)-%x(67)-%c(g)
====> びっくり！
```

これに気づくのに時間が、かかりかなりの時間を無駄にした。

error2

0x21としなければならないところを２１とすると下記のようにエラーが出る。

```
sample0.c:28: error: syntax error before numeric constant
```